Optic Flow from Unstable Sequences through Local Velocity Constancy Maximization

Karl Pauwels *, Marc M. Van Hulle

K.U.Leuven, Laboratorium voor Neuro- en Psychofysiologie,

Herestraat 49–bus 1021, B–3000 Leuven, Belgium

Tel: +32 16 330428 – Fax: +32 16 345960

---

* Corresponding author.

Email addresses: karl.pauwels@med.kuleuven.be (Karl Pauwels),

marc.vanhulle@med.kuleuven.be (Marc M. Van Hulle).

Preprint submitted to Elsevier *28th September 2007*

Abstract

We introduce a novel video stabilization method that enables the extraction of optic flow from short unstable sequences. Contrary to traditional stabilization techniques that use approximative global motion models to estimate the full camera motion, our method estimates the unstable component of the camera motion only. This allows for the use of simpler global motion models, and at the same time extends the validity to more complex environments, such as close scenes that contain independently moving objects. The unstable component of the camera motion is derived from a maximization of the temporal local velocity constancy over the entire short sequence. The method, embedded within a phase-based optic flow algorithm, is tested on both synthetic and complex real-world sequences. The optic flow obtained using our technique is denser than that extracted directly from the original sequence, and from a sequence stabilized with a more traditional stabilization technique.

Key words: multiscale optic flow, video stabilization, phase-based techniques

1 Introduction

Visual motion is a powerful sensory cue used by humans for such diverse purposes as the estimation of self-motion, the extraction of the three dimensional (3D) structure of the environment, and the detection of independently moving objects. This information is crucial for navigation, obstacle avoidance, etc. Due to the ill-posedness of the problem and the presence of external noise influences, extracting the local velocity or optic flow field from an image sequence is difficult. The quality of the estimates can be greatly increased by exploiting the redundancy that is present in a short (e.g. five frames) image sequence. By assuming that the local velocities remain constant over this short sequence, more stable numerical differentiation techniques can be used, temporal aliasing can be reduced, and more reliable confidence measures can be computed [1,2]. If both observer and moving objects undergo smooth motion, this velocity constancy assumption is indeed satisfied in the majority of the scene (except at occlusions). In realistic situations however, shocks and vibrations of the vehicle or robot on which the camera is mounted result predominantly in fast rotational camera movements that induce large local motions over very short time spans [3]. As a result, the local velocities are no longer constant and optic flow algorithms based on this assumption fail to extract meaningful motion vectors. A typical solution is to stabilize the image sequence first.

1.1 Existing Stabilization Techniques

A number of stabilization techniques have been proposed in the past. Since the unstable component of the camera motion is combined with the component

that results from smooth self-motion, traditional stabilization techniques estimate the full camera motion and remove or smooth it afterwards [4]. Camera motion can be decomposed into a 3D translation and a 3D rotation. The local motion field resulting from the translation depends on the scene structure whereas that resulting from the rotation does not. Since both are combined, estimating camera motion in general situations is a nontrivial problem and most algorithms developed for this purpose work well in specific domains only [5]. Some stabilization techniques use a priori knowledge (presence of the horizon, lane markings, the road vanishing point, etc.) to simplify this estimation [3,6]. This limits their applicability to situations where the required features can be reliably obtained. Since shocks and vibrations are mainly 3D rotational, general stabilization techniques typically operate by de-rotating the frames [7], in this way generating a translation-only sequence, or by temporally smoothing the rotational component of the full camera motion [3].

Most stabilization methods rely on simplified global motion models instead (translation; translation, rotation and scaling; affine; quadratic; projective) and only approximate the camera motion [8,9]. For a review of global motion estimation in the context of registration/stabilization, see [10]. Simplified models are only valid in limited scenarios (e.g. aerial imagery) and when they are used in more complex situations (e.g. driving a vehicle downtown or during vehicle turns) the stabilization algorithm typically tracks a dominant component of the background for which the model is sufficiently rich (e.g. the ground plane). Due to the constant changes in the environment however, this dominant component changes also and abrupt changes in the estimated camera motion can result from one frame to another. For this reason, current image stabilization techniques fail when an image contains close scenes [11].

We propose a method that allows estimation of the unstable component of the camera motion only. Since this unstable component consists primarily of 3D rotations, a simple global motion model is sufficient for its estimation. Instead of assuming local velocity constancy, as do multi-frame optic flow algorithms, we maximize it. We thus exploit the fact that smooth camera motion should result in velocity constancy locally in the majority of the scene, irrespective of the complexity of the camera motion, the scene, and the moving objects. By tightly integrating the stabilization and the optic flow computation, the deviations from local velocity constancy can be measured and used to estimate a global 3D rotation for each frame of the short sequence. After correcting for these rotations, the local velocity constancy and the quality of the optic flow increase greatly. By only using 3D rotations in the correction, the component of the flow that results from camera translation is left untouched. Consequently, the flow vectors can still be used in a variety of tasks (egomotion, structure from motion, independent motion, etc. can still be extracted).

The proposed stabilization technique is explained in Section 2 and extensively evaluated on both synthetic and real-world sequences in Section 3. In this evaluation, the algorithm is compared to a traditional stabilization method.

2 Image Sequence Stabilization

Our technique is closely integrated with an existing phase-based optic flow algorithm that we summarize in Section 2.1. This algorithm is particularly suitable for stabilization since it relies on spatial filtering only. The proposed

stabilization method is explained in Section 2.2 and a multiscale extension of the method that allows for large instabilities is discussed in Section 2.3.

2.1 Phase-based Optic Flow using Spatial Filtering

Fleet and Jepson [12] were the first to propose a phase-based technique for the estimation of optic flow. They showed that the temporal evolution of contours of constant phase can yield a good approximation to the local velocity field. The proposed stabilization method is centered around the phase-based optic flow algorithm by Gautama and Van Hulle [2]. The latter method distinguishes itself from [12] by using spatial instead of spatiotemporal filters to compute the phase, and by considering strictly local information when integrating component velocities (normal flow) into full velocities (optic flow). In an extensive comparison, similar to that from [1], the algorithm has been shown to rank among the best ones [2].

For a specific orientation θ, the spatial phase at pixel location $\mathbf{x} = (x, y)^{\mathrm{T}}$ is extracted using 2D complex Gabor filters:

$$G(\mathbf{x}, \mathbf{f}_\theta) = e^{-|\mathbf{x}|^2/\sigma^2} e^{i\,\mathbf{x}\cdot(2\pi\mathbf{f}_\theta)} \ , \tag{1}$$

with peak frequency $\mathbf{f}_\theta = (f_{x,\theta}\,,\, f_{y,\theta})^{\mathrm{T}}$. See Appendix A for a detailed discussion of the filterbank. The filter responses, obtained by convolving the image, $I(\mathbf{x})$, with this oriented filter can be written as:

$$R(\mathbf{x}) = (I * G)(\mathbf{x}) = \rho(\mathbf{x}) e^{i\,\phi(\mathbf{x})} = C(\mathbf{x}) + i\,S(\mathbf{x}) \ . \tag{2}$$

Here $\rho(\mathbf{x}) = \sqrt{C(\mathbf{x})^2 + S(\mathbf{x})^2}$ and $\phi(\mathbf{x}) = \arctan[S(\mathbf{x})/C(\mathbf{x})]$ are the ampli-

Figure 1. about here.

tude and phase components, and $C(\mathbf{x})$ and $S(\mathbf{x})$ are the responses of the quadrature filter pair. The $*$ operator depicts convolution. Points on an equiphase contour satisfy $\phi(\mathbf{x}, t) = c$, with c a constant. Differentiation with respect to time yields:

$$\nabla\phi \cdot \mathbf{v} + \psi = 0 \ , \tag{3}$$

where $\nabla\phi = (\delta\phi/\delta x \, , \, \delta\phi/\delta y)^{\mathrm{T}}$ is the spatial phase gradient, $\mathbf{v} = (v_x, v_y)^{\mathrm{T}}$ is the optic flow vector, and ψ the temporal phase gradient, $\delta\phi/\delta t$. Due to the aperture problem, only the velocity component along the spatial phase gradient can be computed (normal flow). Under a linear phase model, the spatial phase gradient can be substituted by the radial frequency vector, $2\pi\mathbf{f}_\theta$. In this way, the component velocity, $\mathbf{c}_\theta(\mathbf{x})$, can be estimated directly from the temporal phase gradient, $\psi_\theta(\mathbf{x})$:

$$\mathbf{c}_\theta(\mathbf{x}) = -\frac{\psi_\theta(\mathbf{x})}{2\pi|\mathbf{f}_\theta|} \, \frac{\mathbf{f}_\theta}{|\mathbf{f}_\theta|} \ . \tag{4}$$

At each location, the temporal phase gradient is obtained from a linear least-squares fit to the model (see also Fig. 1):

$$\phi_\theta(\mathbf{x}, t) = a + \psi_\theta(\mathbf{x})t \ . \tag{5}$$

The intercept, a, is discarded. A simple unwrapping technique is used to cope with the periodicity of the phase. The reliability of each component velocity is measured by the mean squared error (MSE) of the linear fit, $\sum_t \Big(\Delta\phi_\theta(\mathbf{x}, t)\Big)^2/n$, where n is the number of frames and:

$$\Delta\phi_\theta(\mathbf{x}, t) = \Big(a + \psi_\theta(\mathbf{x})t\Big) - \phi_\theta(\mathbf{x}, t) \ . \tag{6}$$

Each component velocity provides a constraint on the full velocity:

$$|\mathbf{c}_\theta(\mathbf{x})| = \mathbf{v}(\mathbf{x})^{\mathrm{T}} \frac{\mathbf{c}_\theta(\mathbf{x})}{|\mathbf{c}_\theta(\mathbf{x})|} = \mathbf{v}(\mathbf{x})^{\mathrm{T}} \frac{\mathbf{f}_\theta}{|\mathbf{f}_\theta|} \ . \tag{7}$$

If several component velocities with different orientations are present, the corresponding constraints can be combined to estimate the full velocity. Provided a minimal number of component velocities at pixel $\mathbf{x}$ are reliable (their MSE is below a threshold), they are integrated into a full velocity by means of an intersection-of-constraints procedure:

$$\mathbf{v}^*(\mathbf{x}) = \underset{\mathbf{v}(\mathbf{x})}{\operatorname{argmin}} \sum_{\theta \in O(\mathbf{x})} \left(|\mathbf{c}_\theta(\mathbf{x})| - \mathbf{v}(\mathbf{x})^{\mathrm{T}} \frac{\mathbf{c}_\theta(\mathbf{x})}{|\mathbf{c}_\theta(\mathbf{x})|} \right)^2 \ , \tag{8}$$

where $O(\mathbf{x})$ is the set of orientations that correspond to the reliable component velocities. Note that this procedure is strictly local.

2.2 Temporal Phase Gradient Linearization

As mentioned in the introduction, the proposed method estimates a 3D camera rotation for each frame of a short sequence that, when applied to these frames (by warping), maximizes the temporal constancy of the local velocities over the entire short sequence.

The basic idea of the method is illustrated in Fig. 1. Shown in this figure is the temporal sequence of spatial phase (after phase unwrapping) obtained at a certain pixel and for a certain orientation. A line is estimated through these points and the temporal phase gradient, $\psi_\theta(\mathbf{x})$, is obtained. Local velocity constancy is typically reflected in a linear evolution of the phase over time and in small errors in the line-fitting. This is clearly not the case here. The

goal now is to warp the frames in such a way that the deviations from this line (the small arrows) are minimized. The desired changes are computed for each pixel, orientation and frame using Eq. (6). Note that, similar to the temporal phase gradient (Eq. 4), this desired change in the spatial phase can be interpreted as, and transformed into, a component velocity:

$$\Delta \mathbf{c}_\theta(\mathbf{x}, t) = -\frac{\Delta\phi_\theta(\mathbf{x}, t)}{2\pi|\mathbf{f}_\theta|} \frac{\mathbf{f}_\theta}{|\mathbf{f}_\theta|} \ . \tag{9}$$

This component velocity now reflects the local effect (orthogonal to the filter orientation) of the unstable component of the camera motion. Since we know that this component is predominantly 3D rotational [3], its estimation is straightforward. The instantaneous full velocity at pixel location $\mathbf{x}$ that results from a 3D camera rotation, $\boldsymbol{\omega} = (\omega_x, \omega_y, \omega_z)^{\mathrm{T}}$ with ω_p the angular velocity around the p-axis, can be well-approximated by [13]:

$$\mathbf{v}(\mathbf{x}) = B(\mathbf{x})\boldsymbol{\omega} \ , \tag{10}$$

where

$$B(\mathbf{x}) = \begin{bmatrix} xy/f & -f - x^2/f & y \\ f + y^2/f & -xy/f & -x \end{bmatrix} \ , \tag{11}$$

and f the focal length of the camera. For component velocities we have (cf. Eq. 7):

$$|\mathbf{c}_\theta(\mathbf{x})| = \Big(B(\mathbf{x})\boldsymbol{\omega}\Big)^{\mathrm{T}} \frac{\mathbf{c}_\theta(\mathbf{x})}{|\mathbf{c}_\theta(\mathbf{x})|} \ . \tag{12}$$

On the basis of the unstable component velocities, $\Delta\mathbf{c}_\theta(\mathbf{x}, t)$, computed at each pixel, frame and orientation we can now estimate, for each frame, the re-

Figure 2. about here.

quired stabilizing rotation, $\Delta\boldsymbol{\omega}(t)$, by solving the following linear least-squares problem:

$$\Delta\boldsymbol{\omega}^*(t) = \underset{\Delta\boldsymbol{\omega}(t)}{\operatorname{argmin}} \sum_{\mathbf{x},\theta} \left[|\Delta\mathbf{c}_\theta(\mathbf{x},t)| - \Big(B(\mathbf{x})\Delta\boldsymbol{\omega}(t)\Big)^{\mathrm{T}} \frac{\Delta\mathbf{c}_\theta(\mathbf{x},t)}{|\Delta\mathbf{c}_\theta(\mathbf{x},t)|} \right]^2 . \qquad (13)$$

The stabilizing rotations are then used to correct the sequence, and the optic flow is recomputed. The corrections can be done by warping the images or, more efficiently, the Gabor filter outputs (Eq. 2). An overview of the complete stabilization procedure is provided in Figure 2.

In some of the examples of Section 3, the model is further simplified to 2D translations (image shifts), $\Delta\boldsymbol{\tau}(t)$. In this case, Eq. (13) becomes:

$$\Delta\boldsymbol{\tau}^*(t) = \underset{\Delta\boldsymbol{\tau}(t)}{\operatorname{argmin}} \sum_{\mathbf{x},\theta} \left[|\Delta\mathbf{c}_\theta(\mathbf{x},t)| - \Big(\Delta\boldsymbol{\tau}(t)\Big)^{\mathrm{T}} \frac{\Delta\mathbf{c}_\theta(\mathbf{x},t)}{|\Delta\mathbf{c}_\theta(\mathbf{x},t)|} \right]^2 . \qquad (14)$$

Note that not all deviations from linearity in Fig. 1 result from unstable camera motion. Other disturbing factors are image noise, phase singularities, motions exceeding the filter range, etc. These latter errors are however much weaker correlated compared to those resulting from the instabilities. Due to the sheer volume of available measurements, robust and precise rotation estimates can still be obtained. We demonstrate this quantitatively in Section 3.2.

An important limitation of the method discussed in this section is that the magnitude of the effect of the unstable camera motion component has to be within the range of the Gabor filters. To increase this range, and to enable the method to also detect and compensate for large rotational shocks, the stabilization technique has been embedded in a coarse-to-fine multiscale extension of the optic flow algorithm. This is the subject of the next section.

2.3 Multiscale Optic Flow and Stabilization

Due to phase periodicity, phase-based techniques can only detect shifts up to half the filter wavelength. To extend this range, a coarse-to-fine control strategy can be used [14]. An efficient solution involves the use of a Gaussian pyramid [15], in which each level is separated by an octave scale. Starting from the original image resolution at pyramid level $k = 1$, the next level, $k + 1$, is obtained by blurring the images with a Gaussian kernel, $g(\mathbf{x})$, and subsampling:

$$I^{k+1}(\mathbf{x}) = (\mathcal{S}(g * I^k))(\mathbf{x}) \ . \tag{15}$$

The subsampling operator, $\mathcal{S}$, reduces the image resolution to half the resolution of the previous level. The original filters (Eq. 1) are now applied to each level of the pyramid:

$$R^k(\mathbf{x}) = (I^k * G)(\mathbf{x}) \ . \tag{16}$$

By applying the original filters to the lower resolution images, the largest detectable shift effectively doubles at each pyramid level.

The control strategy used in the proposed algorithm is illustrated in Fig. 3. The dashed boxes are specific to the proposed stabilization technique and can be ignored for now. The control strategy starts at the top of the pyramid, level k. Using the optic flow estimate obtained at that resolution, $\mathbf{v}^k$, the phase estimate at the next higher resolution, ϕ^{k-1}, is warped in such a way that the estimated motion is removed [16]:

$$p^{k-1}(\mathbf{x}, t) = \phi^{k-1}\Big(\mathbf{x} - 2 \cdot \mathbf{v}^k(\mathbf{x}) \cdot (3 - t)\,,\, t\Big) \ . \tag{17}$$

Figure 3. about here.

Since the optic flow estimate has been computed at the lower resolution, it needs to be doubled first. The factor $(3-t)$ ensures that each pixel in the five frame sequence ($t = 1, 2, \ldots, 5$) is warped to its corresponding location in the center frame ($t = 3$). Bilinear interpolation is used to perform subpixel warps. Next, the warped phase, p^{k-1}, is used to compute the residual motion. Since a large component of the motion has now been removed, this residual motion is more likely to be within the range of the filters applied to that level. The new optic flow estimate, $\mathbf{v}^{k-1}$, is then obtained by adding the residual motion to $2 \cdot \mathbf{v}^k$. This process is repeated until the pyramid level corresponding to the original image resolution is reached.

The optic flow algorithm we use is particularly suitable for this warping strategy since it uses strictly local information. In our implementation, only optic flow vectors that can be computed reliably (obtained on the basis of a sufficient number of reliable component velocities) at the highest resolution are retained. In other words, if the refinement made at the highest resolution to a lower resolution estimate (that was reliable at that lower resolution) is unreliable, the flow vector is discarded and not included in the density counts of the next section. In this way, overly smooth flow fields are avoided.

A number of additional steps are required to incorporate the proposed stabilization technique in the coarse-to-fine control strategy. They are illustrated by the dashed boxes in Fig. 3. As before, the procedure starts at the lowest resolution. The spatial phase, ϕ^k, is computed at this level and the stabilizing rotations, $\Delta\omega^k$, are estimated as explained in Section 2.2. By applying these

rotations to the estimated phase, the stable phase, q^k, is obtained:

$$q^k(\mathbf{x}, t) = \phi^k\Big(\mathbf{x} + B(\mathbf{x})\Delta\omega^k(t)\,,\, t\Big)\ . \tag{18}$$

The optic flow estimate is then computed on the basis of this stable phase. Compared to the nonstable case (Eq. 17), the phase estimate at the next higher resolution, ϕ^{k-1}, is now warped to also account for the stabilizing rotations at level k:

$$p^{k-1}(\mathbf{x}, t) = \phi^{k-1}\Big(\mathbf{x} - 2 \cdot \mathbf{v}^k(\mathbf{x}) \cdot (3 - t) + 2 \cdot B(\mathbf{x})\Delta\omega^k(t)\,,\, t\Big)\ . \tag{19}$$

Next, the stabilization procedure is applied to this motion-compensated phase, and a refinement of the stabilizing rotations is obtained. Similarly as the optic flow update, the new stabilizing rotations, $\Delta\omega^{k-1}$, are obtained by adding this refinement to $2 \cdot \Delta\omega^k$. Using these updated stabilizing rotations, the phase estimate is rewarped:

$$q^{k-1}(\mathbf{x}, t) = \phi^{k-1}\Big(\mathbf{x} - 2 \cdot \mathbf{v}^k(\mathbf{x}) \cdot (3 - t) + B(\mathbf{x})\Delta\omega^{k-1}(t)\,,\, t\Big)\ , \tag{20}$$

and used to update the optic flow estimate. Finally, the updated rotation and optic flow estimates are propagated to the next level and the procedure is repeated throughout the remainder of the pyramid.

3 Results

In this section, the proposed method is extensively evaluated on synthetic and real-world data. In Section 3.1, the method is compared to a popular alternative stabilization algorithm, on a very difficult but realistic scene. Next, in

Figure 4. about here.

Section 3.2, a synthetic dataset is used to demonstrate the correctness, accuracy and computational efficiency of the method, and finally, in Section 3.3, an extensive comparative evaluation on real-world data is performed.

3.1 Stabilization Example

In this example we demonstrate the advantages of a stabilization approach based on local velocity constancy, over one that attempts to estimate the full camera motion. For this purpose we use a spatiotemporal segment of the widely-used *football* sequence, which is publicly available [1] . The center image of this short sequence (frames 49 up to 53) is shown in Fig. 4A. The players in this scene all move in different directions. Consequently, the sequence contains independent, non-rigid motions that cannot be described by a global model. We remove the surrounding background, since the planar grass field renders the problem trivial for a global method. Note that the remaining segment is very realistic. It can e.g. occur when zooming in on an action sequence, a situation particularly sensitive to camera jitter. Since the original *football* scene is relatively stable, we introduce small instabilities by randomly shifting the individual images horizontally and vertically (up to 4 pixels).

We evaluate the proposed Phase Gradient Linearization method (PGL) in terms of the optic flow density (the percentage of reliable flow vectors) obtained before and after stabilization. A full velocity is considered reliable if the MSE of the linear fit (Eq. 5) does not exceed 0.5 for at least five (out of 11) of the component velocities used in its estimation. Three scales are used in the

[1] `http://www.cipr.rpi.edu/resource/sequences/sif.html`

multiscale implementation of the algorithm. We also evaluate the optic flow density after stabilization with a popular alternative stabilization technique. This technique (TRA) estimates a 2D translation globally by matching the images as a whole [8]. We use the normalized cross correlation measure for reliable matching. In an extensive comparison [8], such a translation-only algorithm showed adequate performance when compared to more sophisticated translation, rotation and scale algorithms. The lack of subpixel matching was identified as the main drawback of the translation-only algorithm. For this reason we have included an additional refinement step in our TRA implementation. Subpixel accuracy is obtained by refining the initial estimate with a gradient-based technique [17]. Central differences are used to estimate the spatial derivatives. This combined procedure enables high-precision image registration. A linearization procedure similar to that shown in Fig. 1 is used to correct the individual 2D translation estimates and to render the estimated camera motion constant over the short sequence. This procedure is explained in more detail in Appendix B.

Figure 4B contains the optic flow field obtained after stabilization with the proposed method, using the 2D translation model (Eq. 14). The flow field is dense and clearly shows all the major motions present in the scene. Figure 4C contains the optic flow field obtained without stabilization. This flow field is much sparser, but the visible flow is largely in accordance with that from Fig 4B. Finally, Fig. 4D shows the optic flow field after stabilization with the alternative method. This flow field is much sparser and different from the other two. This is because, over the course of this short sequence, the global matching algorithm locks onto two different dominant regions. It switches from parts of player 41 to parts of player 29. As a result, the stabilized sequence

does not exhibit a single consistent camera motion. The local velocities are thus also not constant, and the optic flow reliability measure rejects a large proportion of the flow field.

3.2 Stabilization Accuracy and Computational Efficiency

As discussed in Section 2.2, a nonlinear temporal evolution of the spatial phase may also result from disturbances other than the unstable camera motion. In this section we investigate the influence of image layout. We ignore the other two disturbing factors mentioned: image noise and displacements that exceed the filter range. The former can be assumed homogeneous or known, and the latter are handled by the multiscale stabilization. Since the image layout cannot be assumed homogeneous [2], the resulting phase nonlinearity can potentially introduce a directional bias in the estimated stabilizing transformations. In this section, we investigate the presence of such a bias and, at the same time, quantify the stabilization accuracy. We also discuss the computational efficiency of the algorithm. For simplicity, we restrict ourselves to the 2D translational case. Since in the 3D case, the stabilizing rotations are estimated from the same component velocities, the disturbances are similar in that case.

We generate 200 sequences, each five frames in length ($t = 1, 2, \ldots, 5$), by introducing random shifts, $\mathbf{s}(t) = (s_x, s_y)^{\mathrm{T}}$, to a single image (Fig. 5A). The shifts, s_x and s_y, are uniformly distributed between -5 and 5 pixels. Since in this synthetic example, every pixel has the same local motion, an optimally stabilized sequence can be constructed by shifting the images linearly. These

[2] Certain orientations occur more frequently in real-world images.

Figure 5. about here.

'linearized' shifts can be obtained from a linear least-squares regression on the random shifts, $\mathbf{s}(t)$:

$$(\mathbf{s}_t^*, \mathbf{a}^*) = \underset{\mathbf{s}_t, \mathbf{a}}{\operatorname{argmin}} \sum_t \Big((\mathbf{s}_t \cdot t + \mathbf{a}) - \mathbf{s}(t) \Big)^2 \ . \tag{21}$$

It is expected that these optimal shifts are identical to the shifts obtained after stabilization with the proposed method. Our error measure compares both shifts:

$$\boldsymbol{\epsilon}(t) = \Big(\mathbf{s}_t^* \cdot t + \mathbf{a}^* \Big) - \Big(\mathbf{s}(t) + \Delta\boldsymbol{\tau}(t) \Big) \ . \tag{22}$$

We have performed two experiments. In the first, both s_x and s_y are varied, and in the second, only s_x is varied. By fixing s_y to zero in the second experiment, a directional bias becomes more apparent. In both cases, a threshold of 0.01 on the MSE of the linear fit is used to assess the reliability of the component velocities. To facilitate comparison, only the horizontal errors, ϵ_x, are shown in Fig. 5. The errors are averaged over all frames and trials. A very high accuracy can be observed in both experiments. The mean absolute error is $\pm\,0.04$ pixels in both cases. It is also clear from Fig. 5C that the directional bias resulting from the inhomogeneous image structure is very small. It can not be seen in this figure.

The stabilizing translations or rotations estimated by the proposed algorithm are global, and it is thus not necessary to include all pixels and orientations in the estimation (Eqs. 13 and 14). To examine the impact of sample size on performance, we have repeated experiment one, but now using a randomly selected subset of pixels and orientations. On a subset of 0.1% (about 1000

samples), the mean absolute error increases to 0.0382 pixels, as compared to 0.0379 when all data is used. On such a small dataset, the computational overhead of the stabilization method is negligible.

We conclude from this section that the proposed stabilization method is highly accurate in the presence of relatively large translational shifts in all directions. The nonlinear evolution of the phase in certain image regions does not have a large influence on the estimates. This is presumably due to the relatively small image area covered by these regions. Furthermore, the method retains its accuracy when applied to a small, randomly selected subset of the available data. This renders the proposed method highly computationally efficient.

3.3 Real-World Data

In this final experiment, we evaluate the proposed stabilization method (PGL) in terms of the optic flow density obtained on complex real-world driving sequences before and after stabilization. We also include the results obtained after stabilization with the global translation method (TRA) introduced in Section 3.1. Such techniques have been found reasonably effective in stabilizing driving scenes similar to the ones considered here [18]. We again use a threshold of 0.01 on the MSE of the linear fit used to compute the component velocities. As before, at least five reliable component velocities are required. The multiscale implementation of the algorithm uses five frames and three scales.

Both stabilization techniques are applied to two complex real-world driving sequences, recorded in different environments. The sequences have been recorded

Figure 6. about here.

Figure 7. about here.

Table 1

about here.

with a camera rigidly installed behind the front shield of a moving car [3]. The first one, *city*, contains close scenes and relatively small vehicle velocities, and the second sequence, *mway*, involves larger vehicle speeds and also larger destabilizing motions. Independently moving objects are present in both sequences. An example frame of each sequence, together with the optic flow computed for that frame is shown in Fig. 6. It is again clear that the flow computed after stabilization with PGL looks very similar to that computed without stabilization (ORG), except for the greatly increased density. This is because the stabilization procedure averages out the instabilities over the entire short sequence.

Both sequences contain $\pm$ 450 frames of resolution 320×256. The obtained optic flow densities are summarized in Table 1. A two-way ANOVA and Tukey multiple comparison test [19] are used to asses the significance of all individual pairwise differences in mean density at the joint significance level of 0.05. The mean density is underlined in the table if all pairwise differences in which the respective algorithm occurs are significant. This analysis is repeated for each combination of sequence and control strategy (single scale/multiscale). The multiscale strategy improves the density on all occasions. The TRA stabilization technique significantly improves the density as compared to the original sequence, but the proposed method achieves far better results in general and

[3] Courtesy of Dr. Norbert Krüger, University of Southern Denmark, and HELLA Hueck KG, Lippstadt.

in the multiscale scenario in particular.

Figure 7 shows the improvements obtained with PGL in more detail. In this figure the optic flow density is shown as a function of frame number for the entire sequences. In Figs. 7(A–D), the densities obtained without stabilization are shown as black dots and those obtained after stabilization with PGL as solid lines. We can already see significant improvements in the single scale case but the technique fails at certain frames (e.g. around frame 150) in *city* and at various locations in *mway*. The multiscale stabilization overcomes this problem, which clearly shows that large unstable motions are present here (the multiscale results without stabilization are as bad as the single scale at these frames). In the multiscale case, the optic flow density obtained after stabilization is almost constant over the entire sequences. For completeness, the density obtained with TRA is shown in Figs. 7(E,F). Due to the prevalence of close scenes in *city*, the procedure fails often. Better results are obtained on *mway*, but the stabilization remains unreliable and the density is often smaller than that obtained without stabilization.

By repeating the simulations with the proposed method, but now with the simpler 2D translation model (Eq. 14), we have confirmed that the lower optic flow density obtained with TRA does not result from its inability to model rotations around the line of sight. The results obtained with the proposed method were similar for both models. This could either be because unstable rotations around the line of sight are negligible in these sequences, or because rotating (warping) the filter outputs introduces inaccuracies. Since rotations change the orientations, refiltering or a more efficient framework such as steerable filters [20] may be required to further improve the precision. The latter allows for orientation changes without refiltering.

4 Conclusion

We have proposed a novel stabilization technique that does not require estimation of the full camera motion, but instead enables a direct estimation of the unstable component of the camera motion. This is achieved through a maximization of the temporal constancy of the local velocities. The method is computationally efficient as it involves linear systems and simple transformations, the result of which can be computed without time-consuming refiltering. Using synthetic data, we have demonstrated the correctness and accuracy of the method. Although we use a global motion model of similar complexity, we achieve significant increases in reliable optic flow density on real-world sequences as compared to a traditional stabilization technique. It is true that more complex global motion models can be used to more accurately model the camera motion in alternative techniques, but this will be at the cost of efficiency, simplicity, and robustness [21]. Our method on the other hand is simple and valid in the most general of scenes, those where the distance to the scene is small, the range of depths within the scene is large, and moving objects are present.

Acknowledgments

Karl Pauwels and Marc M. Van Hulle are supported by the Excellence Financing program of the K.U.Leuven (EF 2005), the Belgian Fund for Scientific Research – Flanders (G.0248.03, G.0234.04), the Flemish Regional Ministry of Education (Belgium) (GOA 2000/11), the Belgian Science Policy (IUAP P5/04), and the European Commission (NEST-2003-012963, STREP-2002-

Figure A.1. about here.

016276, and IST-2004-027017).

A Filterbank Used to Compute Spatial Phase

Figure A.1 demonstrates the frequency domain coverage of the filters used. The circles correspond to the unit sigma-borders of the enveloping Gaussians. Only the filters indicated by the white circles were used in the original optic flow algorithm [2]. This set of filters covers 11 orientations and two frequency magnitudes $|\mathbf{f}_\theta| = \left\{\frac{1}{12}, \frac{1}{12} \cdot 2^\beta\right\}$. The filter bandwidth, β, is equal to 0.6 octaves, resulting in a spatial width of:

$$\sigma = \frac{2^\beta + 1}{(2^\beta - 1)\, 2\pi\, |\mathbf{f}_\theta|} \ . \tag{A.1}$$

We have extended this filterbank using a pyramid-based approach (see Section 2.3) in which the image resolution is halved at each level and the original filters are applied to these lower resolution images. Three pyramid levels have been used in all the examples. The gray and black circles in Fig. A.1 represent the envelopes when the original filters are applied to images at half, respectively a quarter of the original resolution. It is clear that this procedure increases the frequency domain coverage. The additional low frequency filters enable the detection of larger motion.

We have confirmed experimentally that the results do not critically depend on this particular filterbank. For example, similar results can be obtained with a filterbank that only covers a single frequency magnitude at each pyramid level.

Figure B.1. about here.

B Velocity Constancy Using Frame-to-Frame Translation Estimates

The frame-to-frame translation estimates returned by the matching algorithm, $\mathbf{m}(t) = (m_x, m_y)^{\mathrm{T}}$, indicate the horizontal and vertical translation estimates that transform frame t into frame $t+1$. All estimated global motion can be removed by warping all frames to the center frame of the short sequence. The required transformations are shown in Fig. B.1. In addition to removing the estimated global motion, it is also possible to introduce a constant motion. We have chosen here to introduce a constant motion equal to the average global motion estimated from the original frames, $\bar{\mathbf{m}} = \sum_t \mathbf{m}(t)/4$. This results in the following warps, $\mathbf{w}(t)$, that need to be applied to each frame t:

$$\mathbf{w}(1) = \mathbf{m}(1) + \mathbf{m}(2) - 2 \cdot \bar{\mathbf{m}} \tag{B.1}$$
$$\mathbf{w}(2) = \mathbf{m}(2) - \bar{\mathbf{m}} \tag{B.2}$$
$$\mathbf{w}(3) = 0 \tag{B.3}$$
$$\mathbf{w}(4) = -\mathbf{m}(3) + \bar{\mathbf{m}} \tag{B.4}$$
$$\mathbf{w}(5) = -\mathbf{m}(4) - \mathbf{m}(3) + 2 \cdot \bar{\mathbf{m}} \ . \tag{B.5}$$

This particular choice of constant motion avoids the need for warping in situations where the estimated motion is already constant. In that case, all frame-to-frame translations are equal to the mean and thus all warps are equal to zero. This avoids or reduces (in situations where the estimated motion is nearly constant) the loss of image information near the borders.

References

[1] J. Barron, D. Fleet, S. Beauchemin, Performance of optical flow techniques, International Journal of Computer Vision 12 (1) (1994) 43–77.

[2] T. Gautama, M. Van Hulle, A phase-based approach to the estimation of the optical flow field using spatial filtering, IEEE Trans. Neural Networks 13 (5) (2002) 1127–1136.

[3] Z. Duric, A. Rosenfeld, Shooting a smooth video with a shaky camera, Machine Vision and Applications 13 (5–6) (2003) 303–313.

[4] C. Morimoto, R. Chellappa, Fast electronic digital image stabilization for off-road navigation, Real-Time Imaging 2 (5) (1996) 285–296.

[5] T. Xiang, L. Cheong, Understanding the behavior of SFM algorithms: A geometric approach, International Journal of Computer Vision 51 (2) (2003) 111–137.

[6] Y. Liang, H. Tyan, S. Chang, H. Liao, S. Chen, Video stabilization for a camcorder mounted on a moving vehicle, IEEE Transactions on Vehicular Technology 53 (6) (2004) 1636–1648.

[7] M. Irani, B. Rousso, S. Peleg, Recovery of egomotion using image stabilization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1994, pp. 454–460.

[8] S. Balakirsky, R. Chellappa, Performance characterization of image stabilization algorithms, Real-Time Imaging 12 (2) (1996) 297–313.

[9] Y. Matsushita, E. Ofek, W. N. Ge, X. Tang, H. Y. Shum, Full-frame video stabilization with motion inpainting, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (2006) 1150–1163.

[10] B. Zitova, J. Flusser, Image registration methods: a survey, Image and Vision Computing 21 (2003) 977–1000.

[11] Z. Sun, G. Bebis, R. Miller, On-road vehicle detection: A review, IEEE Transactions on Pattern Analysis and Machine Intelligence 28 (5) (2006) 694–711.

[12] D. Fleet, A. Jepson, Computation of component image velocity from local phase information, International Journal of Computer Vision 5 (1) (1990) 77–104.

[13] G. Adiv, Determining three-dimensional motion and structure from optical flow generated by several moving objects, IEEE Transactions on Pattern Analysis and Machine Intelligence 7 (4) (1985) 384–401.

[14] D. Fleet, A. Jepson, M. Jenkin, Phase-based disparity measurement, CVGIP-Image Understanding 53 (2) (1991) 198–210.

[15] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, J. M. Ogden, Pyramid methods in image processing, RCA Engineer 29 (6) (1984) 33–41.

[16] J. Bergen, P. Anandan, K. Hanna, R. Hingorani, Hierarchical model-based motion estimation, in: European Conference on Computer Vision, 1992, pp. 237–252.

[17] B. Horn, B. Schunck, Determining optical flow, Artificial Intelligence 17 (1-3) (1981) 185–203.

[18] L. Bombini, P. Cerri, P. Grisleri, S. Scaffardi, P. Zani, An evaluation of monocular image stabilization algorithms for automotive applications, in: Proceedings of the IEEE Intelligent Transportation Systems Conference, 2006, pp. 1562–1567.

[19] J. Hsu, Multiple Comparisons: Theory and Methods, Chapman & Hall, London, 1996.

[20] W. Freeman, E. Adelson, The design and use of steerable filters, IEEE Transactions on Pattern Analysis and Machine Intelligence 13 (9) (1991) 891–906.

[21] C. Morimoto, R. Chellappa, Evaluation of image stabilization algorithms, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, Vol. 5, 1998, pp. 2789–2792.

Tables

Table 1

Average flow field density (in percent).

	single scale			multiscale		
seq	ORG	TRA	PGL	ORG	TRA	PGL
city	31.5	37.1	40.1	37.9	44.8	52.2
mway	22.6	26.2	25.8	32.0	32.8	37.1

Figure Captions

Figure 1. Temporal phase gradient linearization. For each orientation and pixel, the temporal phase gradient, $\psi_\theta(\mathbf{x})$, is computed by fitting a line through the spatial phases, $\phi_\theta(\mathbf{x}, t)$, computed at each frame. The proposed stabilization method aims to minimize the deviations from this estimated line, $\Delta\phi_\theta(\mathbf{x}, t)$, by applying a global 3D stabilizing rotation, $\Delta\omega(t)$, to each frame.

Figure 2. Stabilization overview. (**A**) A sliding window (consisting of three frames in this figure) is used to compute optic flow for the central frame, t. (**B**) The spatial phase, ϕ_θ, is computed for each pixel, orientation, θ, (two orientations are shown in this figure) and frame, t. The temporal phase gradient, ψ_θ, is obtained for each pixel and orientation by fitting a linear model to the temporal sequence of the spatial phase. (**C**) The 'unstable' component velocities, $\Delta\mathbf{c}_\theta$, are obtained for each frame and orientation from the deviations between the spatial phases and this linear model. (**D**) A 3D stabilizing rotation, $\Delta\boldsymbol{\omega}(t)$, is estimated for each frame, t, by integrating the 'unstable' component velocities over all pixels and orientations using a linear model. (**E**) These stabilizing rotations define a stabilizing full velocity field for each frame, which is used to warp the images (or the Gabor filter outputs or the phases) and to obtain a stable sequence as a result.

Figure 3. Multiscale optic flow and stabilization. The dashed boxes indicate the additional steps required by the stabilization method. The control strategy starts at the top of the pyramid (level k), corresponding to the lowest resolution images I^k. The phase, ϕ^k, is used to estimate the stabilizing rotations, $\Delta\omega^k$, and warped accordingly. The full velocity, $\mathbf{v}^k$, is then obtained from this

warped phase, q^k. At the next level, $k-1$, both the stabilizing rotations and full velocity are compensated for when warping the phase. Finally, the stabilizing rotations and full velocity are refined by repeating the steps from level k on this warped phase, p^{k-1}, and the process continues until the bottom of the pyramid is reached.

Figure 4. (**A**) Center frame of a spatiotemporal extract from the *football* sequence. Optic flow obtained with (**B**) the proposed stabilization method, (**C**) without stabilization, and (**D**) with the alternative stabilization method (TRA). All flow fields have been subsampled three times.

Figure 5. (**A**) Image used to investigate the stabilization accuracy and a possible directional bias. Difference (in pixels) between the estimated horizontal stabilizing shifts and the optimal shifts when introducing random (**B**) horizontal and vertical shifts, and (**C**) horizontal shifts only.

Figure 6. Example images (**A**) and flow fields (**B**–**E**) obtained on the *city* (top row) and *mway* (bottom row) sequence without stabilization using (**B**) single scale and (**C**) multiscale optic flow, and with the proposed stabilization using (**D**) single scale and (**E**) multiscale optic flow. All flow fields have been subsampled and scaled five times.

Figure 7. Optic flow field density. (**A**–**D**) Results obtained without stabilization (black dots) and after stabilization with the proposed method (solid line) over the entire *city* (left) and *mway* (right) sequences. The first and second row correspond to the results obtained with the single and multiscale algorithm respectively. (**E**,**F**) Results obtained with the alternative stabilization method (TRA) on both sequences using the multiscale implementation.

Figure A.1. Frequency domain coverage of the filterbank from [2] when applied to images at the original resolution (white circles), at half the original resolution (gray circles), and at a quarter of the original resolution (black circles). The circles correspond to the unit sigma-borders of the enveloping Gaussians.

Figure B.1. Transformations required to remove all estimated (translational) global motion from a five frame sequence. All frames are warped to the center frame on the basis of the frame-to-frame translation estimates, $\mathbf{m}(t)$, that transform frame t into frame $t+1$.

Figures

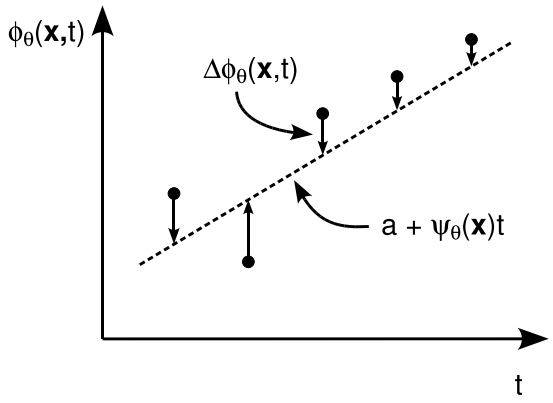

Figure 1.

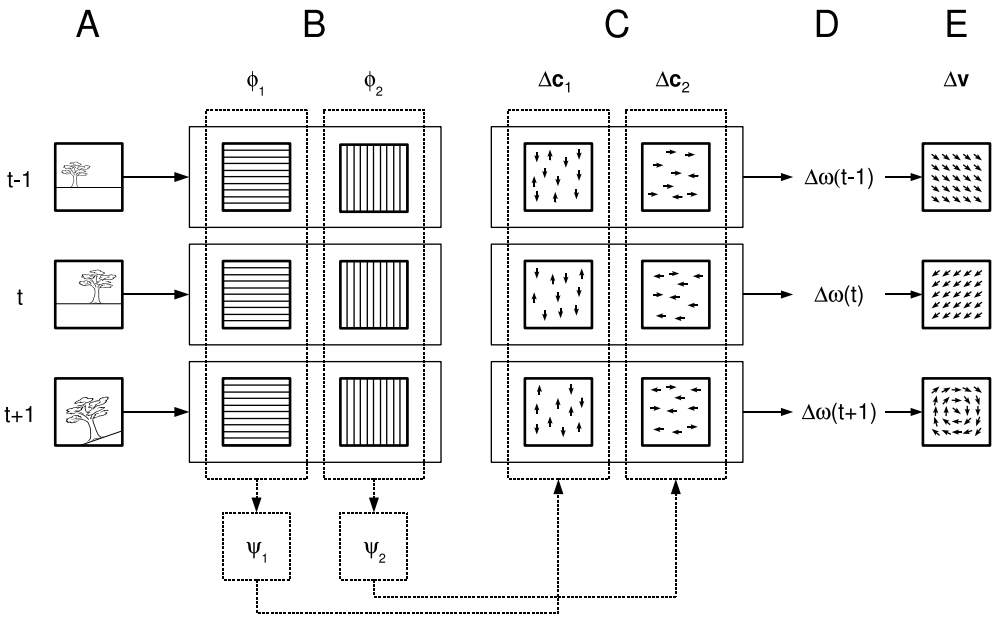
A
B
C
D
E
ϕ_1
ϕ_2
$\Delta\mathbf{c}_1$
$\Delta\mathbf{c}_2$
$\Delta\mathbf{v}$
t-1
t
t+1
$\Delta\omega$(t-1)
$\Delta\omega$(t)
$\Delta\omega$(t+1)
ψ_1
ψ_2

Figure 2.

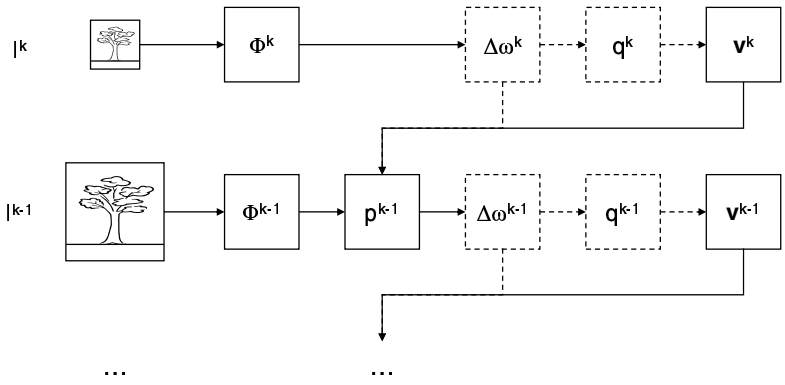

Figure 3.

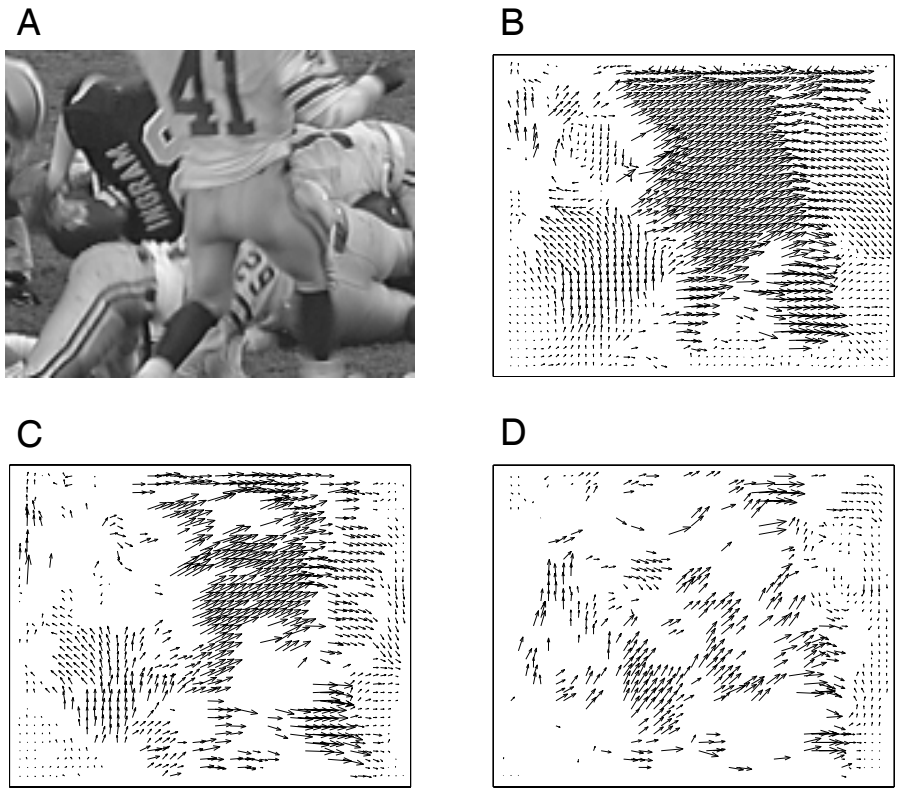

Figure 4.

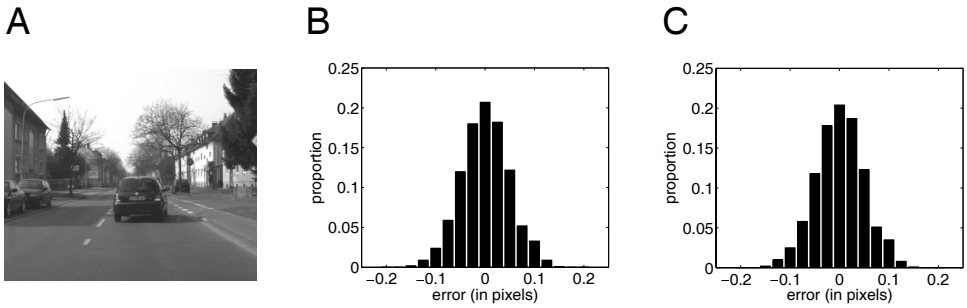

Figure 5.

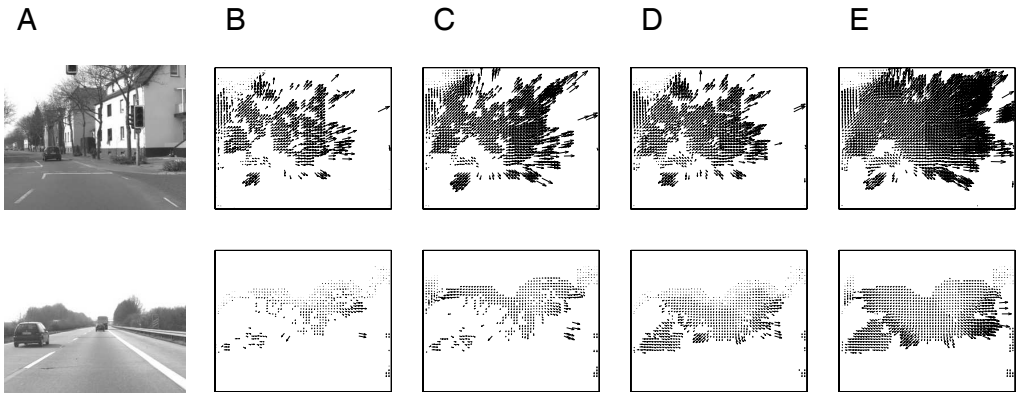

Figure 6.

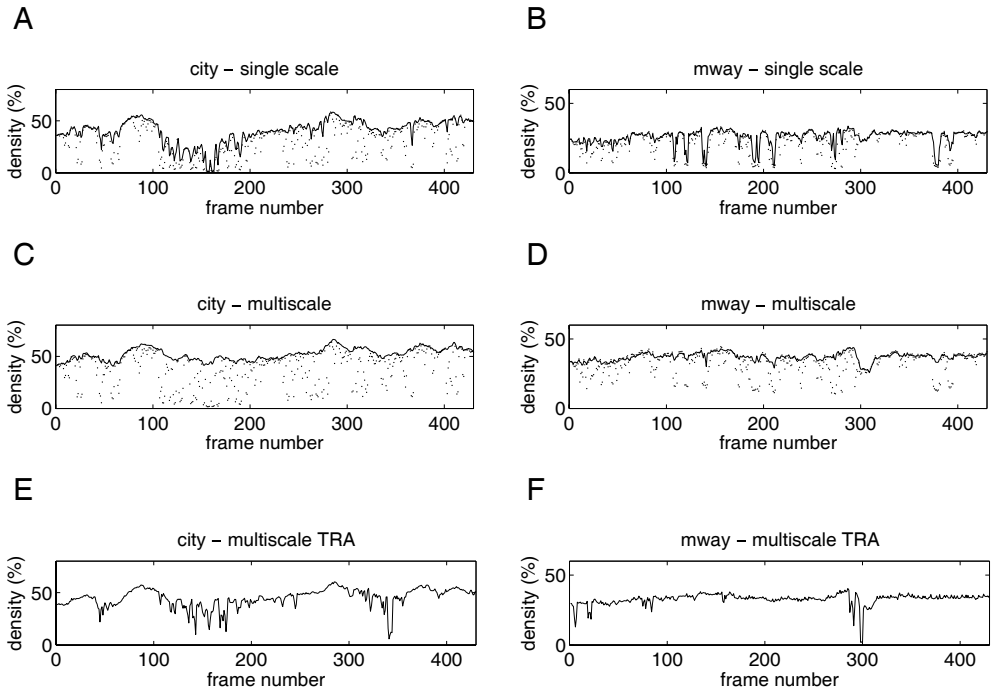
A
city – single scale
density (%)
50
0
0
100
200
300
400
frame number
B
mway – single scale
density (%)
50
0
0
100
200
300
400
frame number
C
city – multiscale
density (%)
50
0
0
100
200
300
400
frame number
D
mway – multiscale
density (%)
50
0
0
100
200
300
400
frame number
E
city – multiscale TRA
density (%)
50
0
0
100
200
300
400
frame number
F
mway – multiscale TRA
density (%)
50
0
0
100
200
300
400
frame number

Figure 7.

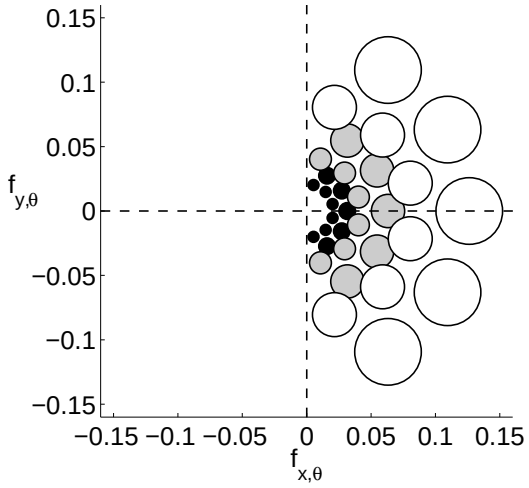

Figure A.1.

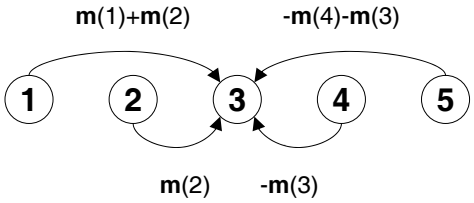

Figure B.1.